

Error Messages Displayed
Miscellaneous Errors

12

Error Messages

This chapter lists the error messages that emanate from standard CP/M and its utility programs. It does not include any error messages from the BIOS; these messages, if any, are the individualized product of the programmers who wrote the various versions of the BIOS.

The error messages are shown in alphabetical order, followed (in parentheses) by the name of the program or CP/M component outputting the message. Messages are shown in uppercase even if the actual message you will see contains lowercase letters. Additional characters that are displayed to “pretty up” the message have been omitted. For example, the message “** ABORTED **” will be listed as “ABORTED”.

Following each message is an explanation and, where possible, some information to help you deal with the error.

The last section of the chapter deals with known errors or peculiarities in CP/M and its utilities. Read this section so that you will recognize these problems when they occur.

Error Messages Displayed

? (CCP)

The CCP displays a question mark if you enter a command name and there is no corresponding “command.COM” file on the disk.

It is also displayed if you omit the number of pages required as a parameter in the SAVE command.

? (DDT)

DDT outputs a question mark under several circumstances. You must use context (and some guesswork) to determine what has gone wrong. Here are some specific causes of problems:

- DDT cannot find the file that you have asked it to load into memory. Exit from DDT and investigate using DIR or STAT (the file may be set to System status and therefore invisible with DIR).
- There is a problem with the data in the HEX file that you have asked DDT to load. The problem could be a bad check-sum on a given line or an invalid field somewhere in the record. Try typing the HEX file out on a console, or use an editor to examine it. It is rare to have only one or two bad bits or bytes in a HEX file; large amounts of the file are more likely to have been corrupted. Therefore, you may be able to spot the trouble fairly readily. If you have the source code for the program, reassemble it to produce another copy of the HEX file. If you do not have the source code, there is no reliable way around this problem unless you are prepared to hand-create the HEX file—a difficult and tedious task.
- DDT does not recognize the instruction you have entered when using the “A” (assemble) command to convert a source code instruction into hexadecimal. Check the line that you entered. DDT does not like tabs in the line (although it appears to accept them) or hexadecimal numbers followed by “H”. Check that the mnemonic and operands are valid, too.

?? = (DDT)

This cryptic notation is used by DDT when you are using the “L” (list disassembled) command to display some part of memory in DDT's primitive assembly language form. DDT cannot translate all of the 256 possible values of a byte. Some of them are not used in the 8080 instruction set. When DDT encounters an untranslatable value, it displays this message as the instruction code, followed by the actual value of the byte in hexadecimal.

You will see this if you try to disassemble code written for the Z80 CPU, which

uses unassigned 8080 instructions. You will also see it if you try to disassemble bytes that contain ASCII text strings rather than 8080 instructions.

ABORTED (STAT)

If you enter any keyboard character while STAT is working its way down the file directory setting files to \$DIR (Directory), \$SYS (System), \$R/W (Read/Write), or \$R/O (Read-Only) status, then it will display this message, stop what it is doing, and execute a warm boot.

By contrast, if you enter the command

```
A>stat *.*<cr>
```

to display all of the files on a disk, there is no way that the process can be aborted.

ABORTED (PIP)

This message is displayed if you press any keyboard character while PIP is copying a file to the list device.

BAD DELIMITER (STAT)

If your BIOS uses the normal IOBYTE method of assigning physical devices to logical devices, you use STAT to perform the assignment. The command has this format:

```
STAT RDR:=PTR:
```

STAT displays this message if it cannot find the “=” in the correct place.

BAD LOAD (CCP)

This is probably the most obscure error message that emanates from CP/M. You will get this message if you attempt to load a COM file that is larger than the transient program area. Your only recourse is to build a CP/M system that has a larger TPA.

BAD PARAMETER (PIP)

PIP accepts certain parameters in square brackets at the end of the command line. This message is displayed if you enter an invalid parameter or an illegal numeric value following a parameter letter.

BDOS ERROR ON d: BAD SECTOR (BDOS)

The BDOS displays this message if the READ and WRITE functions in your BIOS ever return indicating an error. The only safe response to this message is to type CONTROL-C. CP/M will then execute a warm boot. If you type CARRIAGE RETURN, the error will be ignored—with unpredictable results.

A well-implemented BIOS should include disk error recovery and control so that the error will never be communicated to the BDOS. If the BIOS gives you the option of ignoring an error, do so only when you are reasonably sure of the outcome or have adequate backup copies so that you can recreate your files.

BDOS ERROR ON d: FILE R/O (BDOS)

You will see this message if you attempt to erase (ERA) a file that has been set to Read-Only status. Typing any character on the keyboard causes the BDOS to perform a warm boot operation. Note that the BDOS does not tell you *which* file is creating the problem. This can be a problem when you use ambiguous file names in the ERA command. Use the STAT command to display all the files on the disk; it will tell you which files are Read-Only.

This message is also displayed if a program tries to delete a Read-Only file. Again, it can be difficult to determine which file is causing the problem. Your only recourse is to use STAT to try to infer which of the Read-Only files might be causing the problems.

BDOS ERROR ON d: R/O (BDOS)

This looks similar to the previous message, but it refers to an entire logical disk instead of a Read-Only file. However, it is rarely output because you have declared a disk to be Read-Only. Usually, it occurs because you changed diskettes without typing a CONTROL-C; CP/M will detect the new diskette and, without any external indication, will set the disk to Read-Only status.

If you or a program attempts to write any data to the disk, the attempt will be trapped by the BDOS and this message displayed. Typing any character on the keyboard causes a warm boot—then you can proceed.

BDOS ERROR ON d: SELECT (BDOS)

The BDOS displays this message if you or a program attempts to select a logical disk for which the BIOS lacks the necessary tables. The BDOS uses the value returned by SELDSK to determine whether a logical disk “exists” or not.

If you were trying to change the default disk to a nonexistent one, you will have to press the RESET button on your computer. There is no way out of this error.

However, if you were trying to execute a command that accessed the nonexistent disk, then you can type a CONTROL-C and CP/M will perform a warm boot.

BREAK x AT y (ED)

This is another cryptic message whose meaning you cannot guess. The list that follows explains the possible values of “x.” The value “y” refers to the command ED was executing when the error occurred.

x	Meaning
#	Search failure. ED did not find the string you asked it to search for.
?	Unrecognized command.
0	File not found.
>	ED's internal buffer is full.
E	Command aborted.
F	Disk or directory full. You will have to determine which is causing the problem.

CANNOT CLOSE, READ/ONLY? (SUBMIT)

SUBMIT displays this message if the disk on which it is trying to write its output file, "\$\$\$SUB", is physically write protected. Do not confuse this with the disk being *logically* write protected.

The standard version of SUBMIT writes the output file onto the current default disk, so if your current default disk is other than drive A:, you may be able to avoid this problem if you switch the default to A: and then enter a command of the form

```
A><u>submit b:subfile</u><cr>
```

CANNOT CLOSE DESTINATION FILE (PIP)

PIP displays this message if the destination disk is physically write protected. Check the destination disk. If it is write protected, remove the protection and repeat the operation.

If the disk is not protected, you have a hardware problem. The directory data written to the disk is being written to the wrong place, even the wrong disk, or is not being recorded on the medium.

CANNOT CLOSE FILES (ASM)

ASM displays this message if it cannot close its output files because the disk is physically write protected, or if there is a hardware problem that prevents data being written to the disk. See the paragraph above.

CANNOT READ (PIP)

PIP displays this message if you attempt to read information from a logical device that can only output. For example:

```
A><u>pip diskfile=LST:</u><cr>
```

PIP also will display this message if you confuse it sufficiently, as with the following instruction:

```
A><u>pip file1=file2;file3</u><cr>
```

CANNOT WRITE (PIP)

PIP displays this message if you attempt to output (write) information to a logical device that can only be used for input, such as the RDR: (reader, the anachronistic name for the auxiliary input device).

CHECKSUM ERROR (LOAD)

LOAD displays this message if it encounters a line in the input HEX file that does not have the correct check sum for the data on the line.

LOAD also displays information helpful in pinpointing the problem:

```
CHECKSUM ERROR
LOAD ADDRESS 0110 <- First address on line in file
ERROR ADDRESS 0112 <- Address of next byte to be loaded
BYTES READ:
0110:
0110: 00 33 22 2B 02 21 27 02    <- Bytes preceding error
```

Note that LOAD does not display the check-sum value itself. Use TYPE or an editor to inspect the HEX file in order to see exactly what has gone wrong.

CHECKSUM ERROR (PIP)

If you ask PIP to copy a file of type HEX, it will check each line in the file, making sure that the line's check sum is valid. If it is not, PIP will display this message. Unfortunately, PIP does not tell you which line is in error—you must determine this by inspection or recreate the HEX file and try again.

COMMAND BUFFER OVERFLOW (SUBMIT)

SUBMIT displays this message if the SUB file you specified is too large to be processed. SUBMIT's internal buffer is only 2048 bytes. You must reduce the size of the SUB file; remove any comment lines, or split it into two files with the last line of the first file submitting the second to give a nested SUBMIT file.

COMMAND TOO LONG (SUBMIT)

The longest command line that SUBMIT can process is 125 characters. There is no way around this error other than reducing the length of the offending line. You will have to find this line by inspection—SUBMIT does not identify the line.

One way that you can remove a few characters from a command line is to rename the COM file you are invoking to a shorter name, or use abbreviated names for parameters if the program will accept these.

CORRECT ERROR, TYPE RETURN OR CTL-Z (PIP)

This message is a carryover from the days when PIP used to read hexadecimal data from a high-speed paper tape reader. If PIP detected the end of a physical roll

of paper tape, it would display this message. The user could then check to see if the paper tape had torn or had really reached its end. If there was more tape to be read, the user could enter a CARRIAGE RETURN to resume reading tape or enter a CONTROL-Z to serve as the end-of-file character.

Needless to say, it is unlikely that you will see this message if you do not have a paper tape reader.

DESTINATION IS R/O, DELETE (Y/N)? (PIP)

PIP displays this message if you try to overwrite a disk file that has been set to Read-Only status. If you type “Y” or “y”, PIP will overwrite the destination file. It leaves the destination file in Read/Write status with its Directory/System status unchanged. Typing any character other than “Y” or “y” makes PIP abandon the copy and display the message

**** NOT DELETED****

You can avoid this message altogether if you specify the “w” option on PIP’s command line. For example:

```
A>pip destfile=srcfile[lw]<cr>
```

PIP will then overwrite Read-Only files without question.

DIRECTORY FULL (SUBMIT)

This message is displayed if the BDOS returns an error when SUBMIT tries to create its output file, “\$\$\$SUB”. As a rough and ready approximation, use “STAT *.*” to see how many files and extents you have on the disk. Erase any unwanted ones. Then use “STAT DSK:” to find out the maximum number of directory entries possible for the disk.

You may also see this message if the file directory has become corrupted or if the disk formatting routine leaves the disk with the file directory full of some pattern other than E5H.

You can assess whether the directory has been corrupted by using “STAT USR:”. STAT then displays which user numbers contain files. If the directory is corrupt, you will normally see user numbers greater than 15.

It is not easy to repair a corrupted directory. “ERA *.*” erases only the files for the current user number, so you will have to enter the command 16 times, once for each user number from 0 to 15. Alternatively, you can reformat the disk.

DISK OR DIRECTORY FULL (ED)

Self-explanatory.

DISK READ ERROR (PIP)
DISK WRITE ERROR (SUBMIT)
DISK WRITE ERROR (PIP)

These messages will normally be preceded by a BIOS error message. They will only be displayed if the BIOS returns indicating an error. As was described earlier, this is unlikely if the BIOS has any kind of error recovery logic.

END OF FILE, CTL-Z? (PIP)

PIP displays this message if, while copying a HEX file, it encounters a CONTROL-Z (end of file). Again, the underlying idea is based on the concept of physical paper tape. When you saw this message, you could look at the tape in the reader, and if it really was at the end of the roll, enter a CONTROL-Z on the keyboard to terminate the file. Given any other character, PIP would read the next piece of tape.

ERROR : CANNOT CLOSE FILES (LOAD)

LOAD displays this message if you have physically write protected the disk on which it is trying to write the output COM file.

ERROR : CANNOT OPEN SOURCE (LOAD)

LOAD displays this message if it cannot open the HEX file that you specified in the command tail.

ERROR : DISK READ (LOAD)
ERROR : DISK WRITE (LOAD)

These two messages would normally be preceded by a BIOS error message. If your BIOS includes disk error recovery, you would not normally see these messages; the error would have been handled by the BIOS.

ERROR : INVERTED LOAD ADDRESS (LOAD)

LOAD displays this message if it detects a load address less than 0100H in the input HEX file. It also displays the actual address input from the file, so you can examine the HEX file looking for this address to determine the likely cause of the problem.

Note that DDT, when asked to load the same HEX file, will do so without any error—and will probably damage the contents of the base page in so doing.

ERROR : NO MORE DIRECTORY SPACE (LOAD)

Self-explanatory.

ERROR ON LINE N (SUBMIT)

SUBMIT displays this message if it encounters a line in the SUB file that it does not know how to process. Most likely you have a file that has type .SUB but does not contain ASCII text.

The first line of the SUB file is number 001.

FILE EXISTS (CCP)

The CCP displays this message if you attempt to use the REN command to rename an existing file to a name already given to another file.

Use “STAT *.*” to display all of the files on the disk. DIR will show only those files that have Directory status, and you may not be able to see the file causing the problem.

FILE IS READ/ONLY (ED)

ED displays this message if you attempt to edit a file that has been set to Read-Only status.

FILE NOT FOUND (STAT)**FILENAME NOT FOUND (PIP)**

STAT and PIP display their respective messages if you specify a nonexistent file. This applies to both specific and ambiguous file names.

INVALID ASSIGNMENT (STAT)

STAT can be used to assign physical devices to logical devices using the IOBYTE system described earlier. It will display this message if you enter an illogical assignment. Use the “STAT VAL:” command to display the valid assignments.

INVALID CONTROL CHARACTER (SUBMIT)

SUBMIT is supposed to be able to handle a control character in the SUB file—the notation being “^x”, where “x” is the control letter. In fact, the standard release version of SUBMIT cannot handle this notation. A patch is available from Digital Research to correct this problem.

Given that this patch has been installed, SUBMIT will display this message if a character other than “A” to “Z” is specified after the circumflex character.

INVALID DIGIT (PIP)

PIP displays this message if it encounters non-numeric data where it expects a numeric value.

INVALID DISK ASSIGNMENT (STAT)

STAT displays this message if you try to set a logical disk to Read-Only status and you specify a parameter other than "R/O." Note that there is no leading "\$" in this case (as there is when you want to set a file to Read-Only).

INVALID DRIVE NAME (USE A, B, C, OR D) (SYSGEN)

SYSGEN displays this message if you attempt to load the CP/M system from, or write the system to, a disk drive other than A, B, C, or D.

INVALID FILE INDICATOR (STAT)

STAT outputs this message if you specify an erroneous file attribute. File attributes can only be one of the following:

\$DIR	Directory
\$SYS	System
\$R/O	Read-Only
\$R/W	Read/Write

INVALID FORMAT (PIP)

PIP displays this message if you enter a badly formatted command; for example, a "+" character instead of an "=" (on some terminals these are on the same key).

INVALID HEX DIGIT (LOAD)

LOAD displays this message if it encounters a nonhexadecimal digit in the input HEX file, where only a hex digit can appear. LOAD then displays additional information to tell you where in the file the problem occurred:

```
INVALID HEX DIGIT
LOAD ADDRESS 0110  <- First address on line in file
ERROR ADDRESS 0112  <- Address of byte containing non-hex
BYTES READ:
0110:
0110: 00 33      <- Bytes preceding error
```

INVALID MEMORY SIZE (MOVCPM)

MOVCPM displays this message if you enter an invalid memory size for the CP/M system size you want to construct.

INVALID SEPARATOR (PIP)

PIP displays this message if you try to concatenate files using something other than a comma between file names.

INVALID USER NUMBER (PIP)

PIP displays this message if you enter a user number outside the range 0 to 15 with the “[gn]” option (where “n” is the user number).

NO ‘SUB’ FILE PRESENT (SUBMIT)

SUBMIT displays this message if it cannot find a file with the file name that you specified and with a type of .SUB.

NO DIRECTORY SPACE (ASM)**NO DIRECTORY SPACE (PIP)**

Self-explanatory.

NO FILE (CCP)

The CCP displays this message if you use the REN (rename) command and it cannot find the file you wish to rename.

NO FILE (PIP)

PIP displays this message if it cannot find the file that you specified.

NO MEMORY (ED)

ED displays this message if it runs out of memory to use for storing the text that you are editing.

NO SOURCE FILE ON DISK (SYSGEN)

This error message is misleading. SYSGEN does not read source code files. The message should read “INPUT FILE NOT FOUND”.

NO SOURCE FILE PRESENT (ASM)

In this case, ASM really does mean that the source code file cannot be found. Remember that ASM uses a strange form of specifying its parameters. ASM uses the file name that you enter and then searches for a file of that name, but with file type .ASM. The three characters of the file type that you specify are used to represent the logical disks on which the source, hex, and list files, respectively, are to be placed.

NO SPACE (CCP)

The CCP displays this message if you use the SAVE command and there is insufficient room on the disk to accommodate the file.

NOT A CHARACTER SOURCE (PIP)

PIP displays this message if you attempt to copy characters from a character output device, such as the auxiliary output device (known to PIP as PUN:).

OUTPUT FILE WRITE ERROR (ASM)

ASM will display this message if the BDOS returns an error from a disk write operation. If your BIOS has disk error recovery logic, you should never see this message.

PARAMETER ERROR (SUBMIT)

SUBMIT uses the "\$" to mark points where parameter values are to be substituted. If you have a single "\$" followed by an alphabetic character, SUBMIT will display this message. Use "\$\$" to represent a real "\$".

PERMANENT ERROR, TYPE RETURN TO IGNORE (SYSGEN)

SYSGEN displays this message if the BIOS returns an error from a disk read or write operation. If your BIOS has disk error recovery logic, you should never see this message.

QUIT NOT FOUND (PIP)

PIP displays this message when it cannot find the string specified in the "[Qcharacter string^Z]" option, meaning "Quit copying when you encounter this string."

READ ERROR (CCP)

The CCP displays this message if the BIOS returns an error from a disk read or write operation. If your BIOS includes disk error recovery logic, you should not see this error message.

RECORD TOO LONG (PIP)

PIP displays this message if it encounters a line longer than 80 characters while copying a HEX file. Inspect the HEX file using the TYPE command or an editor.

REQUIRES CP/M 2.0 OR NEWER FOR OPERATION (PIP) REQUIRES CP/M VERSION 2.0 OR LATER (XSUB)

Self-explanatory.

SOURCE FILE INCOMPLETE (SYSGEN)

SYSGEN displays this message if the file that you have asked it to read is too short. Use STAT to check the length of the file.

SOURCE FILE NAME ERROR (ASM)

ASM displays this message if you specify an ambiguous file name: that is, one that contains either "*" or "?".

SOURCE FILE READ ERROR (ASM)

ASM displays this message if it encounters problems reading the input source code file. Check the input file using the TYPE command or an editor.

START NOT FOUND (PIP)

PIP displays this message when it cannot find the string specified in the "[Scharacter string^Z]" option, meaning "Start copying when you encounter this string."

SYMBOL TABLE OVERFLOW (ASM)

ASM displays this message when you have too many symbols in the source code file. Your only recourse is to split the source file into several pieces and arrange for ORG (origin) statements to position the generated object code so that the pieces fit together.

SYNCHRONIZATION ERROR (MOVCPM)

Apart from the spelling error, this message is designed to be cryptic. MOVCPM displays it when the Digital Research serial number embedded in MOVCPM does not match the serial number in the version of CP/M that you are currently running.

SYSTEM FILE NOT ACCESSIBLE (ED)

ED displays this message if you attempt to edit a file that has been set to System status. Use STAT to set the file to Directory status.

TOO MANY FILES (STAT)

STAT displays this message if there is insufficient memory available to sort and display all of the files on the specified disk. Try limiting the number of files it has to sort by judicious use of ambiguous file names.

UNRECOGNIZED DESTINATION (PIP)

PIP displays this message if you specify an "illegal" destination device.

VERIFY ERROR (PIP)

If you use the “[v]” (verify) option of PIP when copying to a disk file, PIP will write a sector to the disk, read it back, and compare the data. PIP displays this message if the data does not match.

If there is a problem with your disk system, you should have seen some form of disk error message preceding this one. If there is no preceding message, then you have a problem with the main memory on your system.

Wrong CP/M Version (Requires 2.0) (STAT)

Self-explanatory.

(XSUB ACTIVE) (XSUB)

This is not really an error message, but you may mistake it for one. XSUB is the eXtended SUBMIT program. Without it, SUBMIT can only feed command lines to the Console Command Processor. XSUB allows character-by-character input into any program that uses the BDOS to read console input.

XSUB is initiated by being the first command in a SUB file. Once initiated it stays in memory until the end of the SUB file has been reached. Until that happens, XSUB will output this message every time a warm boot occurs as a reminder that it is still in memory.

XSUB Already Present (XSUB)

XSUB will display this message if it is already active and you attempt to load it again.

Miscellaneous Errors

This section deals with errors that are not accompanied by any error message. It is included here to help you recognize a problem after it has already occurred. The errors are shown grouped by product.

ASM: Fails to Detect Unterminated IF Clause

If you use the IF pseudo-operation, it must be followed by a matching ENDIF. ASM fails to detect the case that the end of the source file is encountered *before* the ENDIF.

If the condition specified on the IF line is false, you could have a situation in which ASM would ignore the majority of the source file without comment.

ASM: Creates HEX File That Cannot Be Loaded

If you omit the `ORG` statement at the front of a source file, ASM will assemble the code originated at location `0000H`. This file will crash the system if you try to load it with DDT. The message “ERROR: INVERTED ADDRESS” will be shown from LOAD.

CP/M: Signs On and Then Dies Without A> Prompt

After the BIOS has signed on, it transfers control to the Console Command Processor. The CCP then attempts to log in the system disk, reading the file directory and building the allocation vector. If your file directory has been badly corrupted, it can cause the system to crash. Use another system disk and try to display the directory on the bad disk.

DDT: Loads HEX File and Then Crashes the System

DDT does not check the addresses specified in a HEX file. If you have forgotten to put an `ORG` statement at the front of the source file, or more subtly, if your source program has “wrapped around” by having addresses up at `0FFFFH` and “above,” the assembler will start assembling at `0000H` again.

DIR: Shows Odd-Looking File Names

If you have odd-looking file names, or the vertical lines of “:” that DIR uses to separate the file names are misaligned, then the file directory has been corrupted. One strategy is to format a new disk, copy all of the valid files to it, and discard the corrupted disk.

DIR: Shows More than One Entry with the Same Name

This can happen if you use a program that creates a new file without asking the BDOS to delete any existing files of the same name. It can also happen if you use the custom `MOVE` utility carelessly.

To remedy the situation proceed as follows:

- Use PIP to copy the specific file to another disk. Do not use an ambiguous file name; specify the duplicated file name exactly. PIP will copy the first instance of the file it encounters in the directory.
- Use the ERA command to erase the duplicated file. *This will erase both copies of the file.*
- Use PIP to copy back the first instance of the file.

STAT: User Numbers > 15

If you use the "STAT USR:" command to display which user numbers contain active files, and user numbers greater than 15 are displayed, then the file directory on the disk has been corrupted.

Use PIP to copy the valid files from legitimate user numbers, and then discard the corrupted disk.

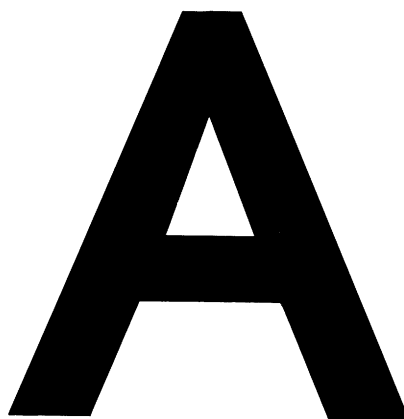
SUBMIT: Fails to Start Submit Procedure

There are several reasons why SUBMIT will not initiate a SUB file:

- You are using the standard release version of SUBMIT and your current default disk is other than drive A:. SUBMIT builds its "\$\$\$SUB" file on the default disk, but the CCP only looks on drive A: for "\$\$\$SUB". Use the following procedure to modify SUBMIT to build its "\$\$\$SUB" file on drive A:

```
A>DDT SUBMIT.COM<cr>
DDT VERS 2.2
NEXT PC
0600 0100
s5bb                <- Change 5bb
05BB 01 00<cr>      <- from 00 (default drive)
05BC 24 .<cr>       to 01 (drive A:)
-^C
A>SAVE 5 SUBMIT.COM<cr>
A>_
```

- If you forgot to terminate the last line of the SUB file with a CARRIAGE RETURN.
- If your SUB file contains a line with nothing but a CARRIAGE RETURN on it (that is, a blank line).



ASCII Character Set

The American Standard Code for Information Interchange (ASCII) consists of a set of 96 displayable characters and 32 nondisplayed characters. Most CP/M systems use at least a subset of the ASCII character set. When CP/M stores characters on a diskette as text, the ASCII definitions are used.

Several of the CP/M utility programs use the ASCII Character Code. Text created using ED is stored as ASCII characters on diskette. DDT, when displaying a “dump” of the contents of memory, displays both the hexadecimal and ASCII representations of memory’s contents.

ASCII does not use an entire byte of information to represent a character. ASCII is a seven-bit code, and the eighth bit is often used for *parity*. Parity is an error-checking method which assures that the character received is the one transmitted. Many microcomputers and microcomputer devices ignore the *parity bit*, while others require one of the following two forms of parity:

Even Parity

The number of binary 1’s in a byte is always an even number. If there is an odd number of 1’s in the character, the parity bit will be a 1; if there is an even number of 1’s in the character, the parity bit is made a 0.

Odd Parity

The number of binary 1’s in a byte is always an odd number. If there is an

even number of 1's in the character, the parity bit will be a 1; if there is an odd number of 1's in the character, the parity bit is made a 0.

Alternative ways of *coding* the information stored by the computer include the 8-bit EBCDIC (Extended Binary Coded Decimal Interchange Code), used by IBM, and a number of *packed binary* schemes, primarily used to represent numerical information.

Table A-1. ASCII Character Codes

				b7 →	0	0	0	0	1	1	1	1
				b6 →	0	0	1	1	0	0	1	1
				b5 →	0	1	0	1	0	1	0	1
b4	b3	b2	b1	Row \ Col.	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	\$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(8	H	X	h	x
1	0	0	1	9	HT	EM)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	
1	1	0	1	13	CR	GS	-	=	M]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	_	o	DEL

NUL	Null	DC1	Device control 1
SOH	Start of heading	DC2	Device control 2
STX	Start of text	DC3	Device control 3
ETX	End of text	DC4	Device control 4
EOT	End of transmission	NAK	Negative acknowledge
ENQ	Enquiry	SYN	Synchronous idle
ACK	Acknowledge	ETB	End of transmission block
BEL	Bell or alarm	CAN	Cancel
BS	Backspace	EM	End of medium
HT	Horizontal tabulation	SUB	Substitute
LF	Line feed	ESC	Escape
VT	Vertical tabulation	FS	File separator
FF	Form feed	GS	Group separator
CR	Carriage return	RS	Record separator
SO	Shift out	US	Unit separator
SI	Shift in	SP	Space
DLE	Data link escape	DEL	Delete

Table A-2. ASCII Character Codes in Ascending Order

Hexadecimal	Binary	ASCII	Hexadecimal	Binary	ASCII
00	000 0000	NUL	30	011 0000	0
01	000 0001	SOH	31	011 0001	1
02	000 0010	STX	32	011 0010	2
03	000 0011	ETX	33	011 0011	3
04	000 0100	EOT	34	011 0100	4
05	000 0101	ENQ	35	011 0101	5
06	000 0110	ACK	36	011 0110	6
07	000 0111	BEL	37	011 0111	7
08	000 1000	BS	38	011 1000	8
09	000 1001	HT	39	011 1001	9
0A	000 1010	LF	3A	011 1010	:
0B	000 1011	VT	3B	011 1011	;
0C	000 1100	FF	3C	011 1100	<
0D	000 1101	CR	3D	011 1101	=
0E	000 1110	SO	3E	011 1110	>
0F	000 1111	SI	3F	011 1111	?
10	001 0000	DLE	40	100 0000	
11	001 0001	DC1	41	100 0001	A
12	001 0010	DC2	42	100 0010	B
13	001 0011	DC3	43	100 0011	C
14	001 0100	DC4	44	100 0100	D
15	001 0101	NAK	45	100 0101	E
16	001 0110	SYN	46	100 0110	F
17	001 0111	ETB	47	100 0111	G
18	001 1000	CAN	48	100 1000	H
19	001 1001	EM	49	100 1001	I
1A	001 1010	SUB	4A	100 1010	J
1B	001 1011	ESC	4B	100 1011	K
1C	001 1100	FS	4C	100 1100	L
1D	001 1101	GS	4D	100 1101	M
1E	001 1110	RS	4E	100 1110	N
1F	001 1111	US	4F	100 1111	O
20	010 0000	SP	50	101 0000	P
21	010 0001	!	51	101 0001	Q
22	010 0010	"	52	101 0010	R
23	010 0011	#	53	101 0011	S
24	010 0100	\$	54	101 0100	T
25	010 0101	%	55	101 0101	U
26	010 0110	&	56	101 0110	V
27	010 0111	'	57	101 0111	W
28	010 1000	(58	101 1000	X
29	010 1001)	59	101 1001	Y
2A	010 1010	*	5A	101 1010	Z
2B	010 1011	+	5B	101 1011	[
2C	010 1100	,	5C	101 1100	\
2D	010 1101	-	5D	101 1101]
2E	010 1110	.	5E	101 1110	^
2F	010 1111	/	5F	101 1111	_

Table A-2. ASCII Character Codes in Ascending Order (Continued)

Hexadecimal	Binary	ASCII	Hexadecimal	Binary	ASCII
60	110 0000		70	111 0000	p
61	110 0001	a	71	111 0001	q
62	110 0010	b	72	111 0010	r
63	110 0011	c	73	111 0011	s
64	110 0100	d	74	111 0100	t
65	110 0101	e	75	111 0101	u
66	110 0110	f	76	111 0110	v
67	110 0111	g	77	111 0111	w
68	110 1000	h	78	111 1000	x
69	110 1001	i	79	111 1001	y
6A	110 1010	j	7A	111 1010	z
6B	110 1011	k	7B	111 1011	{
6C	110 1100	l	7C	111 1100	
6D	110 1101	m	7D	111 1101	}
6E	110 1110	n	7E	111 1110	~
6F	110 1111	o	7F	111 1111	DEL

B

CP/M Command Summary

This appendix summarizes the command line format and the function of each CP/M built-in and transient command. The commands are listed in alphabetical order.

ASM Command Lines

ASM filename<cr> Assembles the file filename.ASM; uses the currently logged disk for all files.

ASM filename.opt<cr> Assembles the file filename.ASM on drive o: (A:,B:,...,P:). Writes HEX file on drive p: (A:,B:,...,P:), or skips if p: is Z:.

Writes PRN file on drive t: (A:,B:,...,P:), sends to console if p: is X:, or skips if p: is Z:.

DDT Command Lines

DDT<cr> Loads DDT and waits for DDT commands.

DDT x:filename.typ<cr> Loads DDT into memory and also loads filename.typ from drive x: into memory for examination, modification, or execution.

DDT Command Summary

- Assss** Enters assembly language statements beginning at hexadecimal address ssss.
- D** Displays the contents of the next 192 bytes of memory.
- Dssss,ffff** Displays the contents of memory starting at hexadecimal address ssss and finishing at hexadecimal address ffff.
- Fssss,ffff,cc** Fills memory with the 8-bit hexadecimal constant cc starting at hexadecimal address ssss and finishing with hexadecimal address ffff.
- G** Begins execution at the address contained in the program counter.
- G,bbbb** Sets a breakpoint at hexadecimal address bbbb, then begins execution at the address contained in the program counter.
- G,bbbb,cccc** Sets breakpoints at hexadecimal addresses bbbb and cccc, then begins execution at the address contained in the program counter.
- Gssss** Begins execution at hexadecimal address ssss.
- Gssss,bbbb** Sets a breakpoint at hexadecimal address bbbb, then begins execution at hexadecimal address ssss.
- Hx,y** Hexadecimal sum and difference of x and y.
- lfilename.typ** Sets up the default file control block using the name filename.typ.
- L** Lists the next eleven lines of assembly language program disassembled from memory.
- Lssss** Lists eleven lines of assembly language program disassembled from memory starting at hexadecimal address ssss.
- Lssss,ffff** Lists the assembly language program disassembled from memory starting at hexadecimal address ssss and finishing at hexadecimal address ffff.

- Mssss,ffff,dddd** Moves the contents of the memory block starting at hexadecimal address ssss and ending at hexadecimal address ffff to the block of memory starting at hexadecimal address dddd.
- R** Reads a file from disk into memory (use “I” command first).
- Rnnnn** Reads a file from disk into memory beginning at the hexadecimal address nnnn higher than normal (use “I” command first).
- Ssss** Displays the contents of memory at hexadecimal address ssss and optionally changes the contents.
- Tnnnn** Traces the execution of (hexadecimal) nnnn program instructions.
- Unnnn** Executes (hexadecimal) nnnn program instructions, then stops and displays the CPU register’s contents.
- X** Displays the CPU register’s contents.
- Xr** Displays the contents of CPU or Flag r and optionally changes them.

DIR Command Lines

- DIR x:<cr>** Displays directory of all files on drive x:. Drive x: is optional; if omitted, the currently logged drive is used.
- DIR x:filename.typ<cr>** Displays directory of all files on drive x: whose names match the ambiguous or unambiguous filename.typ. Drive x: is optional; if omitted, the currently logged drive is used.

DUMP Command Line

- DUMP x:filename.typ <cr>** Displays the hexadecimal representations of each byte stored in the file filename.typ on drive x:. If filename.typ is ambiguous, displays the first file which matches the ambiguous file name.

ED Command Line

- ED x:filename.typ <cr>** Invokes the editor, which then searches for filename.typ on drive x: and creates a temporary file x:filename.\$\$\$ to store the edited text. The filename.typ is unambiguous. Drive x: is optional; if omitted, the currently logged drive is assumed.

ED Command Summary

NOTE: Non-alphabetic commands follow the “Z” command.

- nA** Append lines. Moves “n” lines from original file to edit buffer. 0A moves lines until edit buffer is at least half full.
- +/-B** Begin/Bottom. Moves CP.
 +B moves CP to beginning of edit buffer
 -B moves CP to end of edit buffer.
- +/-nC** Move by characters. Moves CP by “n” character positions.
 + moves forward
 - moves backward.
- +/-nD** Delete characters. Deletes “n” characters before or after the CP in the edit buffer.
 + deletes before the CP
 - deletes after the CP.
- E** End. Ends edit, closes files, and returns to CP/M; normal end.
- nFstring^Z** Find string. Finds the “n”th occurrence of string, beginning the search after the CP.
- H** Move to head of edited file. Ends edit, renames files, and then edits former temporary file.
- I<cr>** Enter insert mode. Text from keyboard goes into edit buffer after the CP; exit with CONTROL-Z.
- Istring^Z** Insert string. Inserts string in edit buffer after the CP.
- Istring<cr>** Insert line. Inserts string and CRLF in the edit buffer after the CP.
- nJfindstring^Zinsertstring^Zendstring^Z** Juxtaposition. Beginning after the CP, finds findstring, inserts insertstring after it, then deletes all following characters up to but not including endstring; repeats until performed “n” times.
- +/-nK** Kill lines. Deletes “n” lines.
 + deletes after the CP
 - deletes before the CP.
- +/-nL** Move by lines. Moves the CP to the beginning of the line it is in, then moves the CP “n” lines forward or backward.
 + moves forward
 - moves backward.
- nMcommandstring^Z** Macro command. Repeats execution of the ED commands in

commandstring “n” times. “n” = 0, “n” = 1, or “n” absent repeats execution until error occurs.

- nNstring^Z** Find string with autoscan. Finds the “n”th occurrence of string, automatically appending from original file and writing to temporary file as necessary.
- O** Return to original file. Empties edit buffer, empties temporary file, returns to beginning of original file, ignores previous ED commands.
- +/-nP** Move CP and print pages. Moves the CP forward or backward one page, then displays the page following the CP. “nP” displays “n” pages, pausing after each.
- Q** Quit edit. Erases temporary file and block move file, if any, and returns to CP/M; original file is not changed.
- R<cr>** Read block move file. Copies the entire block move file X\$\$\$\$\$\$\$.LIB from disk and inserts it in the edit buffer after the CP.
- Rfilename<cr>** Read library file. Copies the entire file filename with extension LIB from the disk and inserts it in the edit buffer after the CP.
- nSfindstring^Zreplacestring^Z** Substitute string. Starting at the CP, repeats “n” times: finds findstring and replaces it with replacestring.
- +/-nT** Type lines. Displays “n” lines.
 + displays the “n” lines after the CP
 - displays the “n” lines before the CP.
- If the CP is not at the beginning of a line
 OT displays from the beginning of the line to the CP
 T displays from the CP to the end of the line
 OTT displays the entire line without moving the CP.
- +/-U** Uppercase translation. After +U command, alphabetic input to the edit buffer is translated from lowercase to uppercase; after -U, no translation occurs.
- OV** Edit buffer free space/size. Displays the decimal number of free (empty) bytes in the edit buffer and the total size of the edit buffer.
- +/-V** Verify line numbers. After +V, a line number is displayed with each line displayed; ED’s prompt is then preceded by the number of the line containing the CP. After -V, line numbers are not displayed, and ED’s prompt is “*”.

- nW** Write lines. Writes first “n” lines from the edit buffer to the temporary file; deletes these lines from the edit buffer.
- nX** Block transfer (Xfer). Copies the “n” lines following the CP from the edit buffer to the temporary block move file X\$\$\$\$\$.LIB; adds to previous contents of that file.
- nZ** Sleep. Delays execution of the command which follows it. Larger “n” gives longer delay, smaller “n” gives shorter delay.
- n:** Move CP to line number “n.” Moves the CP to the beginning of the line number “n” (see “+/-V”).
- :m** Continue through line number “m.” A command prefix which gives the ending point for the command which follows it. The beginning point is the location of the CP (see “+/-V”).
- +/-n** Move and display one line. Abbreviated form of +/-nLT.

ERA Command Lines

- ERA x:filename.typ<cr>** Erases the file filename.typ on the disk in drive x:. The filename and/or typ can be ambiguous. Drive x: is optional; if omitted, the currently logged drive is used.
- ERA x:.*<cr>** Erases all files on the disk in drive x:. Drive x: is optional; if omitted, the currently logged drive is used.

Line Editing Commands

- CONTROL-C** Restarts CP/M if it is the first character in command line. Called *warm start*.
- CONTROL-E** Moves to the beginning of next line. Used for typing long commands.
- CONTROL-H or BACKSPACE** Deletes one character and erases it from the screen (CP/M version 2.0 and newer).
- CONTROL-J or LINE FEED** Same as CARRIAGE RETURN (CP/M version 2.0 and newer).
- CONTROL-M** Same as CARRIAGE RETURN (<cr>).
- CONTROL-P** Turns on the list device (usually your printer). Type it again to turn off the list device.

- CONTROL-R** Repeats current command line (useful with version 1.4); it verifies the line is corrected after you delete several characters (CP/M version 1.4 and newer).
- CONTROL-S** Temporarily stops display of data on the console. Press any key to continue.
- CONTROL-U or CONTROL-X** Cancels current command line (CP/M version 1.4 and newer).
- RUBOUT (RUB) or DELETE (DEL)** Deletes one character and echoes (repeats) it.

Load Command Line

LOAD x:filename<cr> Reads the file filename.HEX on drive x: and creates the executable program file filename.COM on drive x:.

MOVCPM Command Lines

MOVCPM<cr> Prepares a new copy of CP/M which uses all of memory; gives control to the new CP/M, but does not save it on disk.

MOVCPM nn<cr> Prepares a new copy of CP/M which uses “nn” K bytes of memory; gives control to the new CP/M, but does not save it on disk.

MOVCPM * * <cr> Prepares a new copy of CP/M that uses all of memory, to be saved with SYSGEN or SAVE.

MOVCPM nn * <cr> Prepares a new copy of CP/M that uses “nn” K bytes of memory, to be saved with SYSGEN or SAVE.

The “nn” is an integer decimal number. It can be 16 through 64 for CP/M 1.3 or 1.4. For CP/M 2.0 and newer “nn” can be 20 through 64.

PIP Command Lines

PIP<cr> Loads PIP into memory. PIP prompts for commands, executes them, then prompts again.

PIP pipcommandline<cr> Loads PIP into memory. PIP executes the command pip-commandline, then exits to CP/M.

PIP Command Summary

x:new.typ=y:old.typ[p]<cr> Copies the file old.typ on drive y: to the file new.typ on drive x:, using parameters p.

x:new.typ=y:old1.typ[p],z:old2.typ[q]<cr> Creates a file new.typ on drive x: that

consists of the contents of file old1.typ on drive y; using parameters p followed by the contents of file old2.typ on drive z: using parameters q.

x:filename.typ=dev:[p]<cr> Copies data from device dev: to the file filename.typ on drive x:.

dev:=x:filename.typ[p]<cr> Copies data from filename.typ on drive x: to device dev:.

dst:=src:[p]<cr> Copies data to device dst: from device src:.

PIP Parameter Summary

B	Specifies block mode transfer.
Dn	Deletes all characters after the "n"th column.
E	Echoes the copying to the console as it is being performed.
F	Removes form feed characters during transfer.
Gn	Directs PIP to copy a file from user area "n."
H	Checks for proper Intel Hex File format.
I	Ignores any :00 records in Intel Hex File transfers.
L	Translates uppercase letters to lowercase.
N	Adds a line number to each line transferred.
O	Object file transfer (ignores end-of-file markers).
Pn	Issues page feed after every "n"th line.
Qs^Z	Specifies quit of copying after the string "s" is encountered.
R	Directs PIP to copy from a system file.
Ss^Z	Specifies start of copying after the string "s" is encountered.
Tn	Sets tab stops to every "n"th column.
U	Translates lowercase letters to uppercase.
V	Verifies copy by comparison after copy finished.
W	Directs PIP to copy onto an R/O file.
Z	Zeros the "parity" bit on ASCII characters.

PIP Destination Devices

CON:	PUN:	LST:	Logical devices
TTY:	PTP:	LPT:	
CRT:	UPI:	ULI:	
UCI:	UP2:		Physical devices
OUT:	PRN:		Special PIP devices

PIP Source Devices

CON:	RDR:	Logical devices	
TTY:	PTR:		
CRT:	UR1:		
UCI:	UR2:	Physical devices	
NUL:	EOF:	INP:	Special PIP devices

REN Command Line

REN *newname.typ=oldname.typ*<cr> Finds the file *oldname.typ* and renames it *newname.typ*.

SAVE Command Line

SAVE *nnn x:filename.typ*<cr> Saves a portion of the Transient Program Area of memory in the file *filename.typ* on drive *x*: where *nnn* is a decimal number representing the number of pages of memory. Drive *x*: is the option drive specifier.

STAT Command Lines

STAT<cr> Displays attributes and amount of free space for all diskette drives accessed since last warm or cold start.

STAT *x*:<cr> Displays amount of free space on the diskette in drive *x*:

STAT *x:filename.typ*<cr> (**CP/M 2.0 and newer**) Displays size and attributes of file(s) *filename.typ* on drive *x*:. *filename.typ* may be ambiguous. *x*: is optional; if omitted, currently logged drive is assumed.

STAT *x:filename.typ \$atr*<cr> Assigns the attribute *atr* to the file(s) *filename.typ* on drive *x*:. File *filename.typ* may be ambiguous. Drive *x*: is optional; if omitted, currently logged drive is assumed.

STAT **DEV**:<cr> Reports which physical devices are currently assigned to the four logical devices.

STAT **VAL**:<cr> Reports the possible device assignments and partial **STAT** command line summary.

STAT **log:=phy**:<cr> Assigns the physical device *phy*: to the logical device *log*: (may be more than one assignment on the line; each should be set off by a comma).

STAT **USR**:<cr> (**CP/M 2.0 and newer**) Reports the current user number as well as all user numbers for which there are files on currently logged disks.

STAT x:DSK<cr> (CP/M 1.4 and newer) Assigns a temporary write-protect status to drive x:.

SUBMIT Command Lines

SUBMIT filename<cr> Creates a file \$\$\$SUB which contains the commands listed in filename.SUB; CP/M then executes commands from this file rather than the keyboard.

SUBMIT filename parameters<cr> Creates a file \$\$\$SUB which contains commands from the file filename.SUB; certain parts of the command lines in filename.SUB are replaced by parameters during creation of \$\$\$SUB. CP/M then gets commands from this file rather than the keyboard.

SYSGEN Command Line

SYSGEN<cr> Loads the SYSGEN program to transfer CP/M from one diskette to another.

TYPE Command Line

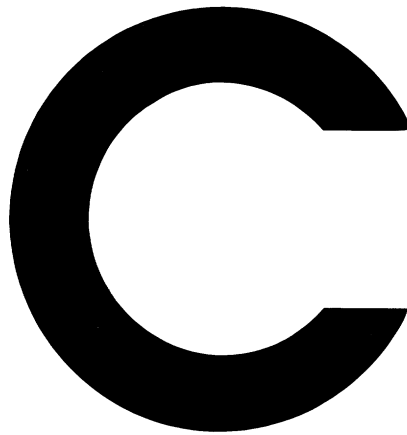
TYPE x:filename.typ<cr> Displays the contents of file filename.typ from drive x: on the console.

USER Command Line

USER n<cr> Sets the User Number to "n," where "n" is an integer decimal number from 0 to 15, inclusive.

x: Command Line

x:<cr> Changes the currently logged disk drive to drive x:. Drive x: can be "A" through "P."



Summary of BDOS Calls

Table C-1. BDOS Function Definitions for CP/M-80 Version 2.2

Function		Entry Parameter(s)	Exit Parameter(s)	Explanation
No.	Name			
00	SYSTEM RESET	None	None	Restarts CP/M-80 by returning control to the the CCP after reinitializing the disk subsystem.
01	CONSOLE INPUT	None	A = ASCII character	Returns the next character typed to the character calling program. Any non-printable character is echoed to the screen (like BACKSPACE, TAB, or CARRIAGE RETURN). Execution does not return to the calling program until a character has been typed. Standard CCP control characters are recognized and their actions performed (CONTROL-P begins or ends printer echoing and so on).

Table C-1. (Continued)

Function		Entry Parameter(s)	Exit Parameter(s)	Explanation
No.	Name			
02	CONSOLE OUTPUT	E = ASCII character	None	Displays the character in the E register on the console device. Standard CCP control characters are recognized and their actions performed (CONTROL-P begins or ends printer echoing and so on.).
03	READER INPUT	None	A = ASCII character	Returns the next character received from the reader device to the calling program. Execution does not return to the calling program until a character is received.
04	PUNCH OUTPUT	E = ASCII character	None	Transmits the character in the E register to the punch device.
05	LIST OUTPUT	E = ASCII character	None	Transmits the character in the E register to the list device.
06	DIRECT CONSOLE IN	E = FF hex	A = ASCII	If register E contains an FF hex, the console device is interrogated to see if a character is ready. If no character is ready, a 00 is returned to the calling program in register A; otherwise the character detected is returned in register A. If register E contains any character other than an FF hex, that character is passed to the console display. All CCP control characters are ignored. The user must protect the program against nonsensical characters being sent from or received by the console device.
	DIRECT CONSOLE OUT	E = ASCII character	None	
07	GET IOBYTE	None	A = IOBYTE	Places a copy of the byte stored at location 0003 hex in the A register before returning control to the calling program.
08	SET IOBYTE	E = IOBYTE	None	Places a copy of the value in register E into the memory location of 0003 hex before returning control to the calling program.
09	PRINT STRING	DE = String address	None	Sends the string of characters stored beginning at the address stored in the DE register pair to the console device. All characters in subsequent addresses are sent until BDOS encounters a memory location which contains a 24 hex (an ASCII "\$"). The CCP control characters are checked for and performed if encountered.

NOTE: CP/M-80 always copies the contents of the H register in the A register if nothing is to be specifically returned in the A register. Some manufacturers, specifically Microsoft, make use of such information to reduce movement of information between the H and A registers.

Table C-1. (Continued)

Function		Entry Parameter(s)	Exit Parameter(s)	Explanation
No.	Name			
0A	READ CONSOLE BUFFER	DE = Buffer address	Data in buffer	This function performs essentially the same as the CCP would in that it takes the characters the user types and stores them into the buffer that begins at the address stored in the DE register pair. The first byte in the buffer pointed to by the DE pair must be the maximum length of the command; BDOS will place the number of characters encountered in the second byte, with the typed command beginning with the third byte pointed to by the DE pair. All standard CCP editing characters are recognized during the command entry.
0B	GET CONSOLE STATUS	None	A = Status	BDOS checks the status of the console device and returns a 00 hex if no character is ready, FF hex if a character has been typed.
0C	GET VERSION NUMBER	None	HL = Version	If the byte returned in the H register is 00 hex then CP/M is present, if 01, then MP/M is present. The byte returned in the L register is 00 if the version is previous to CP/M 2.0, 20 hex if the version is 2.0, 21 hex if 2.1 and so on.
0D	RESET DISK SYSTEM	None		Used to tell CP/M to reset the disk subsystem. Should be used any time diskettes are changed.
0E	SELECT DISK	E = Disk number	None	Selects the disk to be used for subsequent disk operations. A 00 hex in the E register indicates disk A, a 01 hex indicates disk B, etc.
0F	OPEN FILE	DE = FCB address	A = 'Found'/ not found code	Used to activate a file on the current disk drive and current user area. BDOS scans the first 14 bytes of the designated FCB block and attempts to find a match to the filename in the block. A 3F hex (ASCII "?") can be used in any of the filename positions to indicate a "don't care" character. If a match is found, the relevant information about that file is filled into the rest of the FCB by CP/M-80. A value of 00 hex to 03 in register A upon return indicates the open operation was successful, while an FF hex indicates that the file could not be found. If question marks are used to identify a file, the first matching entry is used.

NOTE: CP/M-80 always copies the contents of the H register in the A register if nothing is to be specifically returned in the A register. Some manufacturers, specifically Microsoft, make use of such information to reduce movement of information between the H and A registers.

Table C-1. (Continued)

Function		Entry Parameter(s)	Exit Parameter(s)	Explanation
No.	Name			
10	CLOSE FILE	DE = FCB address	A = 'Found'/ not found code	Performs the opposite of the open file function. A close file function must be performed upon completion of use of any file which has had information written into it.
11	SEARCH FOR FIRST	DE = FCB address	A = 'Found'/ not found code	Performs the same as the open file function with the difference being that the current disk buffer is filled with the 128-byte record which is the directory entry of the matched file.
12	SEARCH FOR NEXT	None	A = 'Found'/ not found code	Performs the same as search for first function except that the search continues on from the last matched entry.
13	DELETE FILE	DE = FCB address	A = 'Found'/ not found code	Changes a flag on the directory entry for the file pointed to by the FCB so that CP/M-80 no longer recognizes it as a valid file. No information is actually erased when this function is performed, although subsequent writes to diskette may use some of the area previously associated with the "deleted" file.
14	READ SEQUEN- TIAL	DE = FCB address	A = Error code	If a file has been activated for use by an open file or make file function, the read sequential function reads the next 128-byte block into memory at the current DMA address. The value of 00 hex is returned in the A register if the read was successful, while any nonzero value in the A register indicates failure.
15	WRITE SEQUEN- TIAL	DE = FCB address	A = Error code	If a file has been activated for use by an open file or make file function, the write sequential function writes the 128-byte block of memory at the current DMA address to the next 128-byte record of the named file.
16	MAKE FILE	DE = FCB address	A = DIR code	Creates a new file with the information (name) indicated by the FCB. CP/M-80 does not check to see if the file indicated already exists, so you must first check to see if the file exists (or delete it). A newly created file need not be opened, as the make file function also performs the necessary opening operations.
17	RENAME FILE	DE = FCB address	A = DIR code	Changes the name of the file referenced by the first 16 bytes of the FCB to the name in the second 16 bytes.

NOTE: CP/M-80 always copies the contents of the H register in the A register if nothing is to be specifically returned in the A register. Some manufacturers, specifically Microsoft, make use of such information to reduce movement of information between the H and A registers.

Table C-1. (Continued)

Function		Entry Parameter(s)	Exit Parameter(s)	Explanation
No.	Name			
18	RETURN LOGIN VECTOR	None	HL = Disk login	The bits in the HL register are used to specify which disk drives are active. The first bit in the L register refers to drive A, the last bit in the H register corresponds to drive P, the highest possible drive. A bit value of 1 indicates active status, a zero denotes an inactive drive.
19	RETURN CURRENT DISK	None	A = Current disk	The numbers 0 through 15 are used to represent the current default disk drive upon return from this function.
1A	SET DMA ADDRESS	DE = DMA	None	Used to select the 128-byte memory block to be used for buffering all disk transfers. Upon system or disk reset, cold or warm start, the buffer is reset to 0080 hex on a normal CP/M-80 system.
1B	GET ALLOC ADDRESS	None	HL = Allocation address	Returns the starting address of the allocation vector, a table which is maintained in memory for each on-line disk drive that indicates the portions of the diskette which are in use.
1C	WRITE PROTECT DISK	None	None	Provides temporary write protection for the diskette in the current default disk drive.
1D	GET R/O VECTOR	None	HL = Disk R/O	Returns a 16-bit value in the HL registers which indicate which drives on the system are write protected. The drives are assigned as in the LOGIN VECTOR, with a value 1 indicating write-protection.
1E	SET FILE ATTRI- BUTES	DE = FCB address	A = DIR code	Sets the file attributes that indicate system/directory and R/O or R/W file status for the file pointed to by the FCB address.
1F	GET DISK PARMS	None	HL = DPB address	Retrieves the disk parameter block for the current active disk drive. These parameters can be used to determine space available on a diskette or to change the characteristics of the disk drive under user control.
20	GET USER CODE SET USER CODE	E = FF E = User code	A = Current User or None	If the E register contains an FF hex, the current user number is returned in the A register. To reset the user number, the appropriate user code is placed in the E register. While the USER command allows user numbers in the range 0-15, this BDOS function can set user numbers in the range of 0-31.

NOTE: CP/M-80 always copies the contents of the H register in the A register if nothing is to be specifically returned in the A register. Some manufacturers, specifically Microsoft, make use of such information to reduce movement of information between the H and A registers.

Table C-1. (Continued)

Function		Entry Parameter(s)	Exit Parameter(s)	Explanation
No.	Name			
21	READ RANDOM	DE = FCB address	A = Error code	Reads the random record number contained in the 33rd, 34th, and 35th byte (a 24-bit address) of the FCB pointed to.
22	WRITE RANDOM	DE = FCB address	A = Error code	Writes information from the current DMA address to the random record pointed to by the number contained in the 33rd, 34th, and 35th bytes of the indicated FCB.
23	COMPUTE FILE SIZE	DE = FCB address	RRF set	Returns the current size of the random record file in the three bytes that constitute the random record field of the FCB. If the third byte contains a 1, then the file contains the maximum record count of 65536, otherwise the value in the first two bytes is a 16-bit value that represents the file size.
24	SET RANDOM RECORD	DE = FCB address	RRF set	Returns the next random record (fills in the random record field of the FCB) after the last sequentially read record. Digital Research suggests that this function is most appropriate to file indexing.
25	RESET DRIVE	DE = Reset drive bits	A = Error code	Forces the specified drives to be reset to the drive bits initial non-logged status.
28	WRITE RANDOM (ZERO)	DE = FCB address	A = Error code	Writes a record of all zeros to diskette before a record is written; useful for identifying unused random records (an unused record would contain zeros instead of data).

NOTE: CP/M-80 always copies the contents of the H register in the A register if nothing is to be specifically returned in the A register. Some manufacturers, specifically Microsoft, make use of such information to reduce movement of information between the H and A registers.

D

Summary of BIOS Calls

Table D-1. CP/M-80 BIOS Routine Definitions

Label in Jump Table	Entry Parameter(s)	Exit Parameter(s)	Explanation
COLDSTART	None	C = 0	Your routine should perform all the necessary start-up operations, including initializing all the values in the base page. Before exiting, the C register must be set to zero.
WARMSTART	None	C = Drive	Your routine should perform all the necessary restart operations but does not need to reinitialize the base page. The C register, on exit, should contain the current drive number.
CONSOLE STATUS (CONST)	None	A = Status	
CONSOLE* INPUT	None	A = Character	

Table D-1. (Continued)

Label in Jump Table	Entry Parameter(s)	Exit Parameter(s)	Explanation
READER* INPUT	None	A = Character	Your routine should wait for a character to be entered at the appropriate device and then return the character in the A register.
CONSOLE* OUTPUT	C = Character	None	
LIST* OUTPUT	C = Character	None	
PUNCH* OUTPUT	C = Character	None	Your routine should take the character in the C register and display it on the appropriate device.
HOME DISK	None	None	The head of the disk drive should be returned to the home position (track 0, sector 0).
SELECT DISK	C = Drive	HL = DHA	Your routine should select the drive indicated by the number in the C register. The HL register on return should contain the address of the disk parameter header.
SET TRACK	C = Track	None	The track indicated by the C register value should be set as the next track to be accessed by the disk drive.
SET SECTOR	C = Sector	None	The sector indicated by the C register value should be set as the next track to be accessed by the disk drive.
SET DMA ADDRESS	BC = DMA address	None	The DMA address indicated by the BC register pair should be set as the address to use for all information transfers from memory to diskette and vice versa.
READ DISK	None	A = Status	Read the current track and sector and transfer the data to the DMA address already set. A 01 hex should be returned if there was an error during transfer.
WRITE DISK	None	A = Status	Write the current track and sector from the data at the DMA address.
SECTOR TRANSLATION	BC = Logical sector DE = Sector map address	HL = Physical sector	A special routine used for systems which maintain data in other than 128-byte blocks. The logical sector on entry is changed to reflect the appropriate actual sector on the diskette.
LIST STATUS	None	A = Status	Your routine should interrogate the appropriate device to see if a character is ready and return a 00 hex in the A register if not ready, or a FF hex if ready.

*All console and device I/O should be done by first looking at the IOBYTE (0003 hex) to determine which device is selected.

Index

A

ANSI Standard Escape sequences:

Support via BIOS, 220

ASCII:

Updating the time in ASCII, 224

ASM:

Assembler, 185

Manual, 6

AUX:

Logical Auxiliary (Reader/Punch) device, 56

Allocation block:

Choosing size, 18

Concepts, 18

In file directory entry, 26

Maximum number in disk parameter block, 34

Prereading used block prior to writing, 155

Reserving in disk parameter block, 35

Allocation vector:

Finding address of, 119

Pointer in disk parameter header, 32

Ambiguous file names:

Avoidance in Rename File, 116

Concepts and restrictions, 24

Example processing, 401

Suggestion for utility program, 426

Used in BDOS Open File, 99

Used in DIR, 50

Used in ERA, 52

Used in Search for First Name Match, 103

Argc, argv:

C Functions for command parameters, 405

Assign:

C program, assigns logical to physical devices, 439

Attributes:

In file directory entry, 26

Available RAM:

Finding amount available, 65

B

BASIC:

Problems with "gobbling" characters, 218

BDOS:

Accessing file directory, C functions, 408

Entry Point, in base page, 59

Errors detected, 296

BDOS Function:

0, System Reset, 71

1, Read Console Byte, 72

2, Write Console Byte, 73

3, Read Reader Byte, 75

4, Write Punch Byte, 77

5, Write List Byte, 77

6, Direct Console I/O, 79

7, Get IOBYTE Setting, 80

8, Set IOBYTE, 86

9, Display \$-Terminated String, 88

10, Read Console String, 90

11, Read Console Status, 94

12, Get CP/M Version Number, 94

13, Reset Disk System, 95

14, Select Logical Disk, 97

15, Open File, 98

16, Close File, 102

17, Search for First Name Match, 103

18, Search for Next File Name Match, 107

19, Erase (Delete) File, 108

20, Read Sequential, 109

21, Write Sequential, 110

22, Create (Make) File, 112

23, Rename File, 115

24, Get Active Disks, 116

25, Get Current Default Disk, 118

26, Set DMA (Read/Write) Address, 118

27, Get allocation vector, 119

28, Set Logical Disk Read-Only, 120

29, Get Read-Only Disks, 120

30, Set File Attributes, 121

31, Get Disk Parameter Block Address, 125

32, Set/Get User Number, 131

33, Read Random, 131

34, Write Random, 133

35, Get File Size, 142

36, Set Random Record Number, 142

37, Reset Logical Disk Drive, 143

40, Write Random with Zero-fill, 144

BDOS Function codes: 69

In LIBRARY.H, 391

Initialization concepts, 12

Interface to other software, 15

Introduction to function calls, 20

Making a function request, 68

Making calls in C, 395

BDOS Function codes (*continued*)

- Naming conventions, 68
- Register conventions for function requests, 70
- Use of Function 0 after hardware error, 299
- Use of Function 0 after printer error, 224
- Use of location 0005H, 14
- What the BDOS does, 67

BDOS Error:

- Bad Sector, 98, 154
- R/O, 120
- Select, 98, 153

BIOS:

- Blocking/Deblocking, 152
- Bootstrap functions, 148
- CONIN, console input, 151
- CONOUT, console output, 151
- CONST, console input status, 150
- Character drivers, debugging, 354
- Components, 147
- Configuration Block, accessing from C, 396
- Debugging, 353
- Debugging interrupts service routines, 357
- Device table, accessing from C, 398
- Different types of disk write, 155
- Direct BIOS calls, example code, 156
- Direct calls, examples, 65
- Direct calls to read/write disk from C, 399
- Disk Parameter Block, accessing from C, 398
- Enhanced BIOS listing, 235
- Enhanced data structures, 225
- Enhancements, 209
- Enhancements to support different protocols, 218
- Entry points, 148
- Example code for standard BIOS, 158
- Feature checklist for debugging, 354
- Finding the jump vector in RAM, 56
- Function key table, accessing from C, 397
- HOME disk heads, 153
- Hardware error handling functions, 296
- Host Buffer, HSTBUF, 152
- Initialization concepts, 12
- Interface to other software, 15
- Jump numbers in LIBRARY.H, 391
- Jump vector, 15, 56
- Keeping the current date, 224
- Keeping the current time, 224
- LIST, list output, 151
- LISTST, list device output status, 156
- Live testing, 368
- Logical Input/Output, 15
- Making calls in C, 396

BIOS (*continued*)

- PUNCH (Auxiliary) output, 151
- Preparing a special version, 184
- READ sector, 154
- READER input, 152
- SECTRAN, logical to physical sector translation, 156
- SELDSK, select disk, 153
- SETDMA, set DMA address, 154
- SETSEC, set sector, 153
- SETTRK, set track, 153
- Sequence of operations for sector write, 155
- Support of function keys, 210
- Using PIP to test, 369
- WRITE sector, 155
- What needs to be tested, 354
- When to avoid direct calls, 15

Backspace:

- CONTROL-H, 47

Bad sector management: 303

- In the BIOS, 154
- Suggestion for utility program, 426, 448

Base page:

- Current user number, 59
- Example memory dumps, 61
- Set by the CCP for loaded program, 54

Basic Debugging for a BIOS: 320

Basic Disk Operating System:

See BDOS

Baud rates:

- Speed, C program to set Baud rates, 431

Bit Bucket:

- If no Punch driver used, 77

Bit map:

See Allocation vector

Bit vector:

- As used in C functions, 402
- Boolean AND, bv_and, Code, 389, Narrative, 404
- Definition of structure in LIBRARY.H, 395
- Display, bv_disp, Code, 389, Narrative, 404
- Fill, bv_fill, Code, 387, Narrative, 404
- Inclusive OR, bv_or, Code, 389, Narrative, 404
- Make, bv_make, Code, 387, Narrative, 404
- Set bit, bv_set, Code, 387, Narrative, 404
- Test bit, bv_test, Code, 388, Narrative, 404
- Test bit non-zero, Code, 388, Narrative, 404

Block mask:

- In disk parameter block, 33

Block shift:

- In disk parameter block, 33

Blocking/Deblocking:

- Concepts, 36

Blocking/Deblocking *(continued)*

Disk write types from BDOS to BIOS, 155
In the BIOS, 152

Bootstrap loader:

Building a new version, 184
Debugging, 351
Example code, 197
Overview, 8

Buffer overflow:

Debugging character driver, 358

Buffer thresholds:

Debugging character driver, 359

Buffer wraparound:

Debugging character driver, 360

Building a new CP/M system:

Example console dialog, 206
The major steps, 183

Building an index file:

Using Set Random Record Number, 143

Building your first CP/M system: 138**Built-in commands:**

In the CCP, 46

Built-in debug code: 321**Bv__and:**

Bit vector, boolean AND, Code, 389, Narrative, 404

Bv__disp:

Bit vector, display, Code, 389, Narrative, 404

Bv__fill:

Bit vector, fill, Code, 387, Narrative, 404

Bv__make:

Bit vector, make, Code, 387, Narrative, 404

Bv__nz:

Bit vector, test bit non-zero, Code, 388, Narrative, 404

Bv__or:

Bit vector, inclusive OR, Code, 389, Narrative, 404

Bv__set:

Bit vector, set bit, Code, 387, Narrative, 404

Bv__test:

Bit vector, test bit, Code, 388, Narrative, 404

C**C Language:**

Reference manuals, 4
Use for utility programs, 371

C programs:

ASSIGN, assigns logical to physical devices, 439
DATE, sets the date, 442
ERASE, a safer way to erase files, 409

C programs *(continued)*

FIND, finds lost files, 416
FUNKEY, sets the function keys, 445
MAKE, makes files visible/invisible, 427
MOVE, moves files between user numbers, 423
PROTOCOL, sets serial line protocols, 434
SPACE, shows used/free disk space, 420
SPEED, sets Baud rates, 431
TIME, sets the time, 442
UNERASE, restores erased files, 412

CBIOS.ASM:

An ingredient for a new system, 185

CCP:

Base page, set for program loaded, 185
Built-in commands, 50
Command Line Editing, 46
Control characters and their effects, 47
Default DMA buffer in base page, 61
Details, 45
ERA, erase (delete) files, 51
Example memory dumps of base page, 61
Functions, 46
Initialization concepts, 12
Interface to other software, 15
Logical devices, 56
Modifying the prompt to show the user number, 235
Overview, 12
Overwriting to gain memory, 45
Program loading, 54
Prompt, 46
REN, rename file, 52
Reloading on warm boot, 45
Resident commands, 14
Returning without warm boot, 66
SAVE, save memory image on disk, 53
Setting of command tail in base page, 60
Setting of default FCB's in base page, 60
TYPE, type an ASCII file, 52
USER, changing user number, 53

CCPM:

Example of Get CP/M Version Number, 95

CDISK:

Example of Reset Disk System, 96

COM file structure: 194**COM files:**

Loaded by the CCP, 46

CON:

Logical console, 16

CONIN:

Accessing the date and time, 223

CONIN (*continued*)

Console input, in the BIOS, 151
Recognizing incoming function key characters, 221
Use with forced input, 219

CONOUT:

Console output, in the BIOS, 151
Escape sequences to input date and time, 223
Processing output escape sequences, 222

CONST:

Console input status, in the BIOS, 50
Problems with programs that "gobble" characters, 218
Use with forced input, 219

CP/M:

Bringing up a new system, 350

CP/M 128-byte "records": 41

CP/M file system:

Concepts, 17

CP/M records as 128-byte sectors: 71

CRC:

See Cyclic Redundancy Check

CRF:

Example of Random Write, 135

Cancel command line:

CONTROL-U, 49

Captions:

For debug subroutines, 322

CARRIAGE RETURN:

CONTROL-M, 48

Changed diskette:

Size of buffer for detection, in disk parameter block, 36

Work area in disk parameter header, 32

Changing disks:

Need to force disk log-in, 96

Changing user number:

USER, 53

Character drivers:

Example testbed, 355

Character I/O:

Enhancements, 213

In the BIOS, 150

Interrupts for input, 215

Practical handling of errors, 299

Choosing allocation block size: 18

Circular buffer:

For interrupt-driven input, 217

Structure in device table, 226

Close File:

BDOS Function 16, 102

Code table:

Definition of structure in LIBRARY.H, 394

Display all strings, ct__disps, Code, 385, Narrative, 407

Get string for code, ct__strc, Code, 386, Narrative, 407

Get string for index, ct__stri, Code, 386, Narrative, 407

Initialize, ct__init, Code, 384, Narrative, 407

Prompt and return code, ct__parc, Code, 384, Narrative, 407

Return code, ct__code, Code, 385, Narrative, 407

Return index, ct__index, Code, 386, Narrative, 407

Used for command tail parameters, 406

Cold Boot:

BIOS functions, 149

Concepts, 12

Command line:

Canceling, CONTROL-U, 49

Deleting last character typed, 49

Repeating, CONTROL-R, 49

Command Line Editing:

By the CCP, 46

Command tail:

Code tables, C functions, 405

Example program to process parameters, 63

In base page, 60

Input to the CCP, 46

Processing, C functions, 405

Communications:

Using Reader/Punch (Auxiliary), 151

Comp__fname:

Compare file name, Code, 374, Narrative, 401

Compare file name:

Comp__fname, Code, 374, Narrative, 401

Configuration Block:

Accessing from C, 396

Concepts, 211

Suggestion for utility program, 448

Variable codes in LIBRARY.H, 391

Console Command Processor:

See CCP

Console output:

From debug subroutines, 323

Temporary pause, CONTROL-S, 47

Console output to printer:

CONTROL-P, 48

Console status:

Debugging character driver, 360

Control characters:

Used in CCP command line editing, 47

Default disk:

Changing, 50
 In base page, 59
 In CCP prompt, 46

Default File Control Blocks:

In base page, 60

Deferred writes:

In conjunction with track buffering, 231

Delete character:

Rubout/Del, 49

Deleting files:

ERA, 51

Device table:

Accessing from C, 398
 Displaying for debugging, 356
 Structure, 225

Digital Research:

Manuals, 6

Direct BIOS calls:

Example code, 156
 Examples, 65
 When to avoid, 15

Directory code:

As returned by BDOS calls, 71
 As returned from Create (Make) File, 114
 As returned from Rename File, 116
 Returned by BDOS Close File, 103
 Returned by BDOS Open File, 99
 Returned by Search for First Name Match, 103
 Returned by Search for Next Name Match, 107

Directory entry: 99

Definition in LIBRARY.H, 394

Directory Parameter Block:

Definition in LIBRARY.H, 393

Disk Drivers:

Debugging, 364

Disk I/O:

Enhancements, 231
 In the BIOS, 152

Disk Map:

In file directory entry, 26

Disk Parameter Block:

Accessing from C, 398
 Adding extra information, 41
 Block shift, mask, and extent mask, 33
 Definition in LIBRARY.H, 394
 Details, 33
 Finding the address of, 125
 Maximum allocation block number, 34

Disk Parameter Block (continued)

Number of directory entries - 1, 35
 Number of tracks before directory, 36
 Pointer in disk parameter header, 31
 Reserving allocation blocks for file directory, 35
 Sectors per track, 33
 Size of buffer for detecting changed diskettes, 36
 Worked example for hard disk, 39

Disk Parameter Header:

Details, 28
 Disk buffer, 31
 Disk parameter block, 31
 Pointer to allocation vector, 32
 Sector skewing, 28
 Work area for changed diskette detection, 32

Disk buffer:

In disk parameter header, 31

Disk definition tables:

Concept, 18
 Details, 27

Disk drivers:

Example testbed code, 365

Disk errors:

Strategy, 303

Disk full:

Error returned from Sequential Write, 112

Disk layout:

CP/M on diskettes, 189

Disk map:

As used in C functions, 402

Disk map clear:

Dm_clr, Code, 382, Narrative, 403

Disk map display:

Dm_disp, Code, 382, Narrative, 403

Diskette:

Layout of standard CP/M diskette, 37

Diskette format:

Concepts, 9

Display S-Terminated String:

BDOS Function 9, 88

Display directory error:

Err_dir, Code, 381, Narrative, 400

Displaying an ASCII file:

TYPE, 52

Displaying current user number: 54**Dm_clr:**

Disk map clear, Code, 382, Narrative, 403

Dm_disp:

Disk map display, Code, 382, Narrative, 403

Control characters (*continued*)

CONTROL-C:

Used to abort after BDOS error, 98

CONTROL-P:

Errors generated, 299

CONTROL-Z:

If no Reader driver in BIOS, 75

Used to indicate end of file, 110

Used to terminate prior to BDOS Close File, 103

Conv__dfname:

Convert directory file name, Code, 375, Narrative, 402

Conv__fname:

Convert file name, Code, 375, Narrative, 408

Convert directory file name:

Conv__dfname, Code, 375, Narrative, 402

Convert file name:

Conv__fname, Code, 375, Narrative, 408

Create (Make) file:

BDOS Function 22, 112

Ct__code:

Code table, return code, Code, 385, Narrative, 407

Ct__disps:

Code table, display all strings, Code, 385, Narrative, 407

Ct__index:

Code table, return index, Code, 386, Narrative, 407

Ct__init:

Code table, initialize, Code, 384, Narrative, 407

Ct__parc:

Code table, prompt and return code, Code, 384, Narrative, 407

Ct__strc:

Code table, get string for code, Code, 386, Narrative, 407

Ct__stri:

Code table, get string for index, Code, 386, Narrative, 407

Current default drive: 97

Current logical disk:

In base page, 59

Current record number:

In FCB, unchanged for Random Read, 132

In FCB, unchanged for Random Write, 132

Current user number:

Displaying, 54

In base page, 59

Customization:

Of CP/M, an overview, 8

Cyclic Redundancy Check:

As used in disk errors, 303

D

DDT:

Dynamic Debug Tool, 185, 329
Manual, 6

I Command used for building new CP/M system, 195

R Command used for building new CP/M system, 195

Used for checking CP/M images, 204

Used for debugging character drivers, 354

Used to create CP/M memory image, 194

Used to debug disk drivers, 364

DESPOOL:

Use of LISTST BIOS entry, 156

DIR:

Display directory of files, 50

DMA buffer:

Default in base page, 60

DPB:

See Disk Parameter Block

DPH:

See Disk Parameter Header

DTR:

PROTOCOL, C program to set protocols, 434

See Data Terminal Ready

Data storage area:

Concept, 17

Data Terminal Ready:

Explanation of DTR protocol, 219

DATE:

C program, sets the date, 442

Date:

Keeping the current date in the BIOS, 224

Reading the date from the console driver, 223

Debug output:

Controlling when it occurs, 324

Debug subroutines: 322

Overall design philosophy, 322

Debugging a new CP/M system, 319

Debugging checklist:

Character output, 361

Disk drivers, 367

Interrupt service routines, 359

Non-interrupt service routine, 359

Real Time Clock, 362

Default DMA Address: 118

Default DMA buffer:

In base page, 60

- DO:**
Suggestion for utility program, 448
- DPB:**
See Disk Parameter Block
- DPH:**
See Disk Parameter Header
- E**
- ED:**
Editor, manual, 6
- ERA:**
Erase (delete) files, 51
- Echoing of keyboard characters:**
Read Console Byte, 72
- End of File:**
Detection using Read Sequential, 110
- Erase (Delete) File:**
BDOS Function 19, 108
- ERASE:**
C program, a safer way to erase files, 409
- Erased files:**
Unerasing them, 26
- Erasing a file:**
ERA, 51
Logical deletion only, 23
- Err_dir:**
Display directory error, Code, 381, Narrative, 400
- Error messages:**
Debugging disk drivers, 368, Chapter 12
- Errors:**
Dealing with hardware errors, 295
Example printer error routine, 301
Handling disk errors, 303
Hardware, analysis, 297
Hardware, correction, 299
Hardware, detection strategy, 296
Hardware, indication, 297
Improved disk error messages, 312
Practical handling, character I/O, 299
- Escape sequences:**
Function keys, debugging character driver, 360
Incoming, debugging character driver, 360
Processing output sequences, 222
Recognizing function key sequences, 222
Suggestion for utility program, 448
Support via device table, 226
- EtX/Ack:**
Debugging character drivers, 358, 362
Explanation of protocol, 219
- EtX/Ack (continued)**
Protocol, C program to set protocols, 434
- Example programs:**
Ordering diskette, 4
- Extent:**
In file directory entry, 26
Of files, concepts, 18
- Extent mask:**
In disk parameter block, 33
- F**
- FCB:**
Default FCB's in base page, 60
See File Control Block
- FDOS:**
Rarely used term for BDOS/CCP combined
- File Attributes:** 99
Setting, 121
See File status
- File Control Block:**
Creating one from an ASCII file name, 100
Concepts, 18
Definition in LIBRARY.H, 393
Structure, 41
Used for random file operations, 43
Used for sequential file operations, 43
Used in BDOS Open File, 99
Used in BDOS Requests, 71
- File Directory:**
Accessing entries directly, 399
Processing, C functions, 402
- File Organizations:**
Concepts, 41
- File Protection:**
Special characters in file name, 114
- File changed:**
File status bit in file directory entry, 26
- File directory:**
Accessing, C functions, 400
Accessing, via BDOS & C functions, 408
Concept, 17
Details, 18
Disk map, 26
Displaying contents, DIR, 50
Entry structure, 22
Erasing files, ERA, 51
File extent, 26
File name and type in entry, 27
Matching names, C functions, 401
Number of entries — 1, in disk parameter block, 35

File directory (*continued*)

Number of tracks before, 36

Record number, 27

Status (attribute) bits, 26

User number in entry, 22

File extent:

Concepts, 18

In file directory entry, 26

Manipulation to achieve Random I/O, 110-12

Opening extent 0 for Random I/O, 133-34

File name/type:

In file directory entry, 23

File protection:

Suggestion for utility program, 426

File status:

In file directory entry, 26

File system:

Concepts, 17

File type:

Conventions for actual types, 24

Filecopy:

Suggestion for utility program, 426

Files:

Creating, sequence of operations, 20

Displaying a directory, DIR, 50

Find:

C program, finds lost files, 416

Flushing buffers:

Prior to BDOS Close File, 103

Forced input:

Concepts, 219

Debugging character driver, 360

Suggestion for utility program, 448

Framing error:

Character I/O, handling, 300

Function Key table:

Accessing from C, 397

Function keys:

Structure in LIBRARY.H, 392

Support with enhanced BIOS, 220

Testing in a live BIOS, 370

FUNKEY:

C program, sets the function keys, 445

G

GETC:

Example of Read Sequential, 111

GETDPB:

Example of Get Disk Parameter Block Address, 126

GFA:

Example of Get File Attributes, 122

GNF:

Example of Search First/Next File Name Match, 104

Get CP/M Version Number:

BDOS Function 12, 94

Get Current Default Disk:

BDOS Function 25, 118

Get Disk Parameter Block Address:

BDOS Function 31, 125

Get Disk Parameter Block Address:

Get__dpb, Code, 383

Get File Size:

BDOS Function 35, 142

Get IOBYTE Setting:

BDOS Function 7, 80

Get Read-Only Disks:

BDOS Function 29, 120

Get allocation vector:

BDOS Function 27, 119

Get configuration block address:

Get__cba, 372

Get next directory entry:

Get__nde, Code, 378, Narrative, 400

Get next file name:

Get__nfn, Code, 376, Narrative, 408

Get__cba:

Get configuration block address, 372

Get__dpb:

Get Disk Parameter Block Address, Code, 383

Get__nde:

Get next directory entry, Code, 378, Narrative, 400

Get__nfn:

Get next file name, Code, 376, Narrative, 408

H

HEX file structure: 195

HOME:

Home disk heads, in the BIOS, 153

HSTBUF:

In the BIOS, 152

Hard disk:

Division into several logical disks, 39

Special considerations, 36

Hardware errors:

Dealing with, 295, Chapter 9

Hardware reset:

Debugging character driver, 359

Heath/Zenith:

Special version of CP/M, 55

Host Buffer:

In the BIOS, 152

Host sector size:

In the BIOS, 152

I**I/O Redirection:**

Assign, C program to assign physical devices, 439

Concepts, 214

IOBYTE Structure, 57

IF/ENDIF directives:

Used for debug subroutines, 323

IOBYTE:

Equates for bit fields, 86

Structure, 57

Use for polling communications line, 75

Use with Direct Console I/O for communications, 80

Initialization of debug subroutines: 323**Input redirection:**

Debugging character driver, 359

Input/Output:

Fake I/O for debugging purposes, 327

Interactions:

Between CCP, BDOS, and BIOS, 15

Interlace:

See Sector skewing

Interrupt service routines:

Debugging checklist, 357

Interrupts:

Architecture, 216

Circular buffers, 217

Dealing with buffer overflow, 219

Debugging service routines, 329

Use for character input drivers, 215

J**Johnson-Laird Inc.:**

Ordering diskette, 4

Jump vector:

Use for entering the BIOS, 15

L**LIBRARY.C:**

Utility function library, 372

LIBRARY.H:

Header for LIBRARY.C functions, 390

LIST:

List output, in the BIOS, 151

LISTST:

List device output status, in the BIOS, 156

LST:

Logical list device, 56

Line editing:

Using Read Console String, 91

Line feed:

CONTROL-J, 48

List Device Errors:

Problems with BDOS Function 5, 78

Loading CP/M:

Overview, 11

Loading programs:

Via the CCP, 54

Loadsys:

Suggestion for utility program, 448

Location 0000H:

Use for warm boot, 13

Location 0005H:

Simple examples of use, 20

Use for BDOS function calls, 14

Logging in a disk:

Using BDOS Reset Disk System, 96

Logical deletion of files, 23

ERA, 51

Logical devices:

CON:, LST:, AUX:, RDR:, PUN:, 56

Logical disk:

As represented in File Control Block, 42

Division of hard disk into several logical disks, 39

Selecting, 97

Logical Input/Output:

As afforded by the BIOS, 15

Logical records:

Concepts, 41

Logical sectors to physical: 28

SECTTRAN, in the BIOS, 156

Login Vector:

See BDOS Function 24, 116

Lowercase letters in file name: 114**M-disk:**

Using memory as an ultra-fast disk, 232

M80:

Macro Assembler, 185

MAC:

Macro Assembler, 185

MAKE:

C program, makes files visible/invisible, 427

MOVE:

C program, moves files between user numbers, 423

MOVCPM:

In conjunction with patches to CP/M, 234

Relocating the CCP and BDOS, 201

Use in building a new CP/M system, 182

MSGOUT:

Example of Write Console Byte, 74

MSGOUTI:

Example of Write Console Byte, 74

Manuals:

From Digital Research, 6

Maximum allocation block number:

In disk parameter block, 34

Memory:

Displaying in debug subroutines, 324

Finding size of area available for programs, 65

Use of hidden memory for buffers, 216

Used as an ultra-fast disk, 232

Memory dumps:

Base page, 61

Memory image:

Checking a new system, 204

Of new CP/M system, 185

Memory layout:

For example BIOS, 190

For input to SYSGEN, 187

With CP/M loaded, 13

Messages:

As an aid to debugging, 326

N

Notation:

For example console dialog, 3

Number of file directory entries:

In disk parameter block, 35

O

OM:

Example of Display \$-Terminated String, 89

OPENF:

Example of Open File, 100

Open File:

BDOS Function 15, 98

Open directory:

Open_dir, Code, 378, Narrative, 400

Open_dir:

Open directory, Code, 378, Narrative, 400

Orville Wright approach to debugging: 320

Output Escape sequence:

Debugging character output driver, 362

Overrun error:

Character I/O, handling, 300

Overwriting the CCP:

To gain memory, 45

Owner:

Suggestion for utility program, 426

P

PIP:

Used to test a new BIOS, 369

PROM Bootstrap:

Used to load CP/M, 11

PUN:

Logical Punch, 56

PUNCH:

Punch (Auxiliary) output, in the BIOS, 151

PUTC:

Example of Write Sequential, 113

PUTCPM:

Example program, 191

Writing a utility, 189

Parallel printers:

Error handling, 301

Parameters:

Example program to process command tail, 63

Parity error:

Character I/O, handling, 300

Pass counters:

Use in debug subroutines, 324

Patching CP/M:

General techniques, 234

Performance:

Effect of sector skewing, 29

Physical end of line:

CONTROL-E, 47

Physical sectors:

Relative, on a hard disk, 38

Polled Reader Input:

Problems and solutions, 75

Polled communications:

Using Direct Console I/O, 80

Printer echo:

CONTROL-P, 48

Printer errors:

Example routine, 301

Use of watchdog timer, 224

Printer timeout error:

Handling, 300

Program loading:

Via the CCP, 54

Program termination:

Returning to CP/M, 66

Prompt:

From the CCP, 46

Protect/Unprotect:

Suggestion for utility program, 426

PROTOCOL:

C program, sets serial line protocols, 434

Protocol:

See also Data Terminal Ready, Request to Send, Xon/Xoff, Etx/Ack

Definitions in LIBRARY.H, 392

Support in enhanced BIOS, 218

Support via device table, 226

Xon/Xoff, used by TYPE, 52

Public files:

Patches to create this feature, 235

Suggestion for utility program, 448

Public/Private:

Suggestion for utility program, 448

R**RAM-disk:**

Using memory as an ultra-fast disk, 232

RCS:

Example of Direct Console I/O, 81

RDR:

Logical Reader, 56

READ:

Read Sector, in the BIOS, 154

READER:

Reader input, in the BIOS, 152

REN:

Rename file, 52

RF:

Example of Rename File, 117

RLSRDR:

Example of Read Reader Byte, 76

RMAC:

Relocatable Macro Assembler, 185

RO:

Example of Random File I/O, 136

RSA:

Example of Read Console String, 92

RST7:

Use for debugging drivers, 356

RTS:

See also Buffer thresholds, Request to Send Protocol, C program to set protocols, 434

Random Read:

Using Read Sequential, 110

Random Write:

Using Write Sequential, 112

Random files:

Concepts, 43

Creating an empty file, 144

Problem of sparse files, 44

Virtual size, 142

Random record number:

In FCB, set for Random Read, 132

In FCB, set for Random Write, 132

Rd_disk:

Read disk (via BIOS), Code, 377, Narrative, 400

Read Console Byte:

BDOS Function 1, 72

Read Console Status:

BDOS Function 11, 94

Read Console String:

BDOS Function 10, 90

Read Random:

BDOS Function 33, 131

Read Reader Byte:

BDOS Function 3, 75

Read Sequential:

BDOS Function 20, 109

Read disk (via BIOS):

Rd_disk, Code, 377, Narrative, 400

Read-Only:

Automatic setting after changing diskettes, 32

File status bit in file directory entry, 26

Read-Only Disks: 120**Read-Only File:**

Attribute bit, 121

Read/write directory:

Rw_dir, Code, 380, Narrative, 400

Reading/Writing disk:

Direct BIOS calls from C, 399

Real Time Clock:

Debugging, 362

Example testbed code, 363

TIME, C program to set the time, 444

Reclaim:

Suggestion for utility program, 426

Record number:

In file directory entry, 26

Manipulation to achieve Random I/O, 110, 112

Registers:

Displaying in debug subroutines, 324

Relative page offset:

Use for making direct BIOS calls, 65

Relative physical sectors:

On a hard disk, 38

Release diskettes:

Files from Digital Research, 6

Rename File:

BDOS Function 23, 115

Renaming a file:

REN, 52

Repeat command line:

CONTROL-R, 48

Request to Send:

Explanation of RTS protocol, 219

Reserved area:

Concept, 17

Reset:

Signal used to start loading of CP/M, 11

Reset Disk System:

BDOS Function 13, 95

Reset Logical Disk Drive:

BDOS Function 37, 143

Resident CCP commands: 14

Restoring registers:

In interrupt service routine, 356

Rw_dir:

Read/write directory, Code, 380, Narrative, 400

S

SAVE:

Save memory image in disk file, 53

Use in building new CP/M system, 194

SECTRAN:

Logical sector to physical, in the BIOS, 156

SELDSK:

Debugging disk drivers, 367

Select disk, in the BIOS, 153

SETDMA:

Set DMA Address, in the BIOS, 154

SETSEC:

Set Sector, in the BIOS, 153

SETTRK:

Set Track, in the BIOS, 153

SETTRK/SEC:

Debugging disk drivers, 367

SFA:

Example of Set File Attributes, 122

SID:

Debugging tool, 330

STAT:

Use for displaying current user number, 54

SYSGEN:

System Generator, 185

Writing a new system to disk, 186

Savesys:

Suggestion for utility program, 448

Saving memory on disk:

SAVE, 53

Search First/Next:

Example use together, 107

Search for file:

Srch_file, Code, 376, Narrative, 408

Search for Next File Name Match:

BDOS Function 18, 107

Require for Search for First, 104

Sector interlace:

See Sector skewing

Sector size:

Host, in the BIOS, 152

Sector skewing:

Effect on performance, 29

For CP/M image on disk, 190

In disk parameter header, 28

Sector skipping:

Concepts, 304

Sector sparing:

Concepts, 304

Sectors:

Use in allocation blocks, 18

Sectors per track:

In disk parameter block, 33

Select Logical Disk:

BDOS Function 14, 97

Sequential Files:

Concepts, 43

Set DMA (Read/Write) Address:

BDOS Function 26, 118

Required by Search for First Name Match, 104

- Set File Attributes:**
BDOS Function 30, 121
- Set IOBYTE:**
BDOS Function 8, 86
- Set Logical Disk Read-Only:**
BDOS Function 28, 120
- Set Random Record Number:**
BDOS Function 36, 142
- Set disk parameters for rd/wrt__disk:**
Set__disk, Code, 378, Narrative, 400
- Set search control block:**
Setscb, Code, 381, Narrative, 401
- Set/Get User Number:**
BDOS Function 32, 131
- Set__disk:**
Set disk parameters for rd/wrt__disk, Code, 378, Narrative, 401
- Setscb:**
Set search control block, Code, 381, Narrative, 401
- Setterm:**
Suggestion for utility program, 448
- Shadow PROM:**
Used to load CP/M, 11
- Short:**
Minor change to C Language, 395
- Single-density, single-sided:**
Diskette format, 10
- Single disk reset, 143**
- Skewing:**
See Sector skewing
- Skipping:**
Skipping bad sectors on disk, 304
- SPACE:**
C program, shows used/free disk space, 420
- Spare:**
Suggestion for utility program, 448
- Spare directory:**
Debugging disk drivers, 367
- Sparing:**
Use of spare sectors on disk, 304
- Sparse Random Files:**
Problem, 44
- Special version of CP/M:**
Heath/Zenith, 55
- SPEED:**
C program, sets baud rates, 431
- Srch__file:**
Search for file, Code, 376, Narrative, 408
- Sstrcmp:**
Substring compare, 373
- Stack:**
Filling with known pattern, 323
- Stack overflow:**
In interrupt service routine, 358
- Standard BIOS:**
Example code, 158
- String scan:**
Strscn, 372
- String scan, uppercase:**
Ustrscn, 372
- Strscn:**
String scan, 372
- Structure:**
Of CP/M, 5
- Subroutine:**
CCPM, Check if CP/M Version 2, 95
CDISK, Change Disk, 96
CRF, Create Random File, 135
DB\$Blank, Display a blank, 344
DB\$CAH, Convert A to ASCII Hex., 343
DB\$CRLF, Display Carriage Return, Line Feed, 344
DB\$Colon, Display a colon, 344
DB\$Conin, Debug console input, 336
DB\$Conout, Debug console output, 336
DB\$DAH, Display A in Hex., 343
DB\$DHLH, Display HL in Hex., 343
DB\$Display\$CALLA, Display call address, 343
DB\$Display, Main debug display, 338
DB\$GHV, Get Hex. Value, 348
DB\$Init, Debug initialize, 335
DB\$Input, Debug Port Input, 346
DB\$MEMORY, Debug display of memory/registers, 325
DB\$MESSG, Display Message, 345
DB\$MSGI, Display Message (In-line), 345
DB\$Off, Turn debug output off, 337
DB\$On, Turn debug output on, 337
DB\$Output, Debug Port Output, 347
DB\$Pass, Decrement the pass counter, 337
DB\$Set\$Pass, Set pass counter, 337
DIVHL, Divide HL by DE, 129
FOLD, Fold lowercase to upper, 93
FSCMP, Folded String Compare, 93
GAB, Get Allocation Block given Track/Sector, 128
GDTAS, Get Directory Track/Sector, 127
GETC, Get Character from Sequential File, 111
GETDPB, Get Disk Parameter Block Address, 126
GFA, Get File Attributes, 122
GMTAS, Get Maximum Track/Sector, 127

Subroutine (*continued*)

GNF, Get Next File matching ambiguous name, 104
GNTAS, Get Next Track/Sector, 128
GTAS, Get Track/Sector from Allocation block No., 126
MSGOUT, Message Output, 74
MSGOUTI, Message Output In-Line, 74
MULHL, Multiply HL by DE, 129
OM, Output Message selected by A register, 89
OPENF, Open File given ASCII file name, 100
PUTC, Put Character to Sequential File, 113
RCS, Read Console String, 81
RF, Rename File, 117
RL\$RDR, Read Line from Reader, 76
RO, Random File I/O (non-128-byte records), 136
RSA, Return Subprocessor Address, 93
SDLR, Shift DE,HL one bit right, 141
SFA, Set File Attributes, 122
SHLR, Shift HL right one bit, 130
SUBHL, Subtract DE from HL, 130
TERM, Terminal Emulator, 87
TOUPPER, Fold lowercase to upper, 84
WL\$LST, Write Line to List Device, 79
WL\$PUN, Write Line to Punch, 78

Substring compare:

Sstrcmp, 373
Uppercase: Usstrcmp, 373

System file:

Attribute bit, 121
File status bit in file directory entry, 26
Not displayed by DIR, 51

System Reset:

BDOS Function 0, 71

T

TERM:

Example of Set/Get IOBYTE, 87

TIME:

C program, sets the time, 442

TYPE:

Type an ASCII file, 52

Tab:

Interaction of tab characters and escape sequences, 222

Tab expansion:

Supported by Write Console Byte, 73
Using Display \$-Terminated String, 89

Termination of programs, returning to CCP: 45

Testbed:

Use for new drivers, 353

Time:

Correct display during debugging, 364
Keeping the current time in the BIOS, 224
Reading the time from the console driver, 223

Top of RAM:

Finding, via base page, 60

Track buffering:

Enhancement to disk I/O, 231

Track offset:

See Tracks before directory

Tracks before directory:

In disk parameter block, 36

Transient Program Area:

Finding available size, 65

Typeahead:

Concepts, 217
Dealing with buffer overflow, 219

U

Undo command line:

CONTROL-U, 49

UNERASE:

C program, restores erased files, 412

User Number:

Changing under program control, 131
Changing using USER, 53
Displaying, 54
In base page, 59
In file directory entry, 22
Patches to make this appear in CCP prompt, 235
Suggestion for utility program, 426

Usstrcmp:

Uppercase substring compare, 373

Ustrcmp:

Uppercase string scan, 372

Utility programs: 371

V

Variable record lengths:

Processing in Random Files, 133, 134

W

WL\$LST:

Example of Write List Byte, 79

WL\$PUN:

Example of Write Punch Byte, 78

WRITE:

Write sector, in the BIOS, 155

Warm Boot:

After BDOS Error, 98

Warm Boot (*continued*)

BIOS functions, 150
 Initiated by CONTROL-C, 47
 Initiated by pressing a key, 94
 Initiated by System Reset BDOS Function, 72
 JMP at location 0000H, 55
 Reloading the CCP, 45
 Resetting Read-Only disks, 120
 Setting default DMA Address, 118
 Technique for avoiding, 66
 Use of location 0000H, 13

Watchdog timer:

Concepts, 225
 Debugging Real Time Clock, 364
 Use for detecting printer errors, 224

Write Console Byte:

BDOS Function 2, 73

Write List Byte:

BDOS Function 5, 77

Write Punch Byte:

BDOS Function 4, 77

Write Random:

BDOS Function 34, 133

Write Random with Zero-fill:

BDOS Function 40, 144

Write Sequential:

BDOS Function 21, 110

Write disk (via BIOS):

Wrt__disk, Code, 377, Narrative, 400

Wrt__disk:

Write disk (via BIOS), Code, 377, Narrative, 400

X**Xoff:**

CONTROL-S, 48

Xon:

CONTROL-Q, 49

Xon/Xoff:

Debugging character driver, 358, 362

Explanation of protocol, 240

PROTOCOL, C program to set protocols, 434

Supported by Read Console Byte, 72

Use by TYPE, 53

Z**ZSID:**

Z80 Symbolic Interactive Debugger, 185, 350